

Overview of scenario tree generation methods, applied in financial and economic decision making

Miklós Vázsonyi

Received 2007-09-14

Abstract

Scenario tree generation methods are powerful decision-making tools when decisions have to be made under uncertainty. Instead of giving a point estimation of multivariate random variables scenario tree generation methods provide likely scenarios of future with associated probabilities. The scenarios can cover only the next time step or even more steps ahead in time. This paper summarizes the most common scenario tree generation methods applied in financial prediction to give an overall overview of the field. Application examples are provided to illustrate real-life implementations.

Keywords

scenario tree generation · financial decision making · probabilistic programming · default prediction · time series prediction

1 Introduction to scenario tree generation

In models of decision making under uncertainty we face the problem of representing the uncertainties in a form which is suitable for quantitative models. If the uncertainties are expressed in terms of multivariate continuous distributions, or a discrete distribution with far too many outcomes, we normally face two possibilities: either creating a decision model with internal sampling, or try to find a simple discrete approximation of the given distribution that serves as input to the model [13]. There is a rich literature about modelling and estimation of continuous-state financial processes, but little attention has been paid to approximate such a process by a discrete-state scenario process and how to measure the pertaining approximation error [2]. This is why scenario tree generation methods gained importance in previous years. These tools have the power to support decision making under uncertainty. During scenario tree generation we do not forecast the future state of a random variable but try to generate a finite set of realistic possible scenarios [1]. The prediction of multivariate financial and economic time series can be modelled by multistage stochastic programs. These models use a finite set of scenarios and corresponding probabilities to model the multivariate random data process. A scenario tree represents the abstract structure of scenarios. A simple example is illustrated in the scenario tree of Figs. 1, 2.

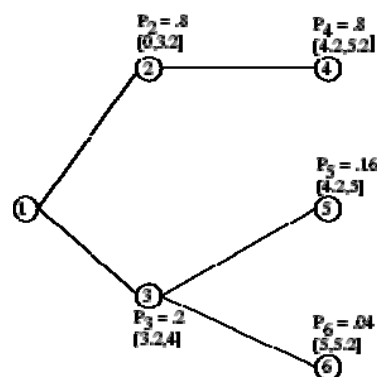


Fig. 1. Example of simple scenario tree

Each complete path from the root node n_1 to one of the leaves n_6, \dots, n_{10} represents a scenario, here the tree consists of five

Miklós Vázsonyi

Department of Information and Knowledge Management, BME, H-1521 Budapest, Műegyetem rkp. 3., Hungary
 e-mail: miklos@vazsonyi.com

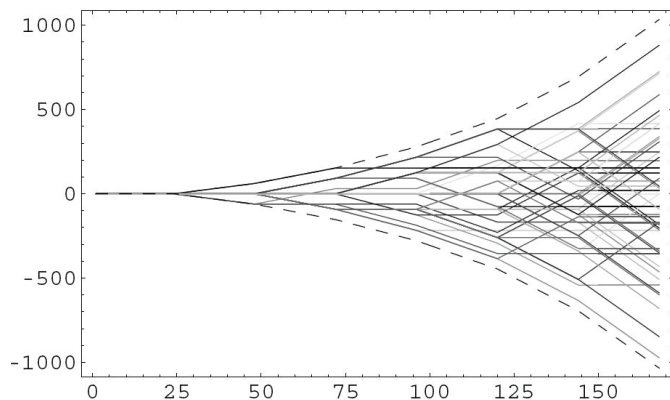


Fig. 2. Example of realistic scenario tree

scenarios. Each scenario corresponds to a particular outcome of the random quantity at hand. Scenarios are realizations (trajectories) of a certain multidimensional stochastic process. The scenarios and their associated probabilities form a discrete approximation of the probability distribution of the data process. Approximations of stochastic processes in form of scenario trees are useful for the formulation of multiperiod dynamic decision models as multistage stochastic programs. A multistage stochastic programming model will determine an optimal decision for each node of the scenario tree, given the information available at that point [3].

Stochastic programming requires a coherent representation of uncertainty. This is expressed in terms of a multivariate continuous distribution. Hence, a decision model is generated with internal sampling or a discrete approximation of the underlying continuous distribution. A method to obtain the discrete outcomes for the random variables is referred to as scenario tree generation. In multistage models, at each time period, new scenarios branch from old, creating a scenario tree. The random variables are the uncertain return values of each asset on an investment. The discretization of the random values and the probability space leads to a framework in which a random variable takes finitely many values. Thus, the factors driving the risky events are approximated by a discrete set of scenarios, or sequence of events. Given the event history up to a particular time, the uncertainty in the next period is characterized by finitely many possible outcomes for the next observation. This branching process is represented using a scenario tree. The root node in the scenario tree represents the 'today' and is immediately observable from deterministic data. The nodes further down represent the events of the world which are conditional at later stages. The arcs linking the nodes represent various realizations of the uncertain variables. An ideal situation is that a generated set of scenarios represents the whole universe of possible outcomes of the random variable. Therefore, scenarios should include both optimistic and pessimistic projections [14].

A detailed model framework is described in paper [8] to communicate and create scenario generation systems. To structure the process, they suggest five key questions to be answered.

Then a framework is introduced to describe, discuss and implement scenario tree generation simply and consistently. The proposed framework consists of four main components and several secondary components, allowing the process to be communicated effectively to disparate audiences. For more details see paper [8].

2 Known methods of scenario tree generation

2.1 Bootstrapping historical data

The simplest approach for generating scenarios is to use only the available data without any mathematical modelling. It bootstraps a set of historical records. Each scenario is a sample of assets returns which is obtained by sampling returns that were observed in the past. Dates from the available historical records are selected randomly and for each date in the sample we read the returns of all asset classes or risk factors during the month prior to that date. These are scenarios of monthly returns. If we want to generate scenarios of returns for along horizon – say 1 year – we sample 12 monthly returns from different points in time. The compounded return of the sampled series is the 1-year return. With this approach the correlations among asset classes are preserved. [15] Bootstrapping is a proper method for estimating the sampling distribution of an estimator by resampling with replacement from the original sample. There are more complicated bootstraps for sampling without replacement, two-sample problems, regression, time series, hierarchical sampling, mediation analyses, and other statistical problems. Bootstrapping is becoming the most popular method of testing mediation because it does not require the normality assumption to be met, and because it can be effectively utilized with smaller sample sizes ($N < 20$). Usage of this method is suggested when sample size is small and we would not like to make any normality assumptions.

3 Vasicek model

A very simple way to simulate the time evolution of financial time series (e.g.: interest rates) relies on the Vasicek model whose main characteristic is the presence of a mean-reverting term and a constant volatility term. From a mathematical viewpoint it is a stationary Gaussian Markovian model. The Brownian motion with unit variance and the other variables of the Vasicek formula are constant model parameters which are estimated by looking at historical market data. There are many methods to estimate the value of these model parameters, like Method of Moments and the Maximum Likelihood method are widely used to this purpose [1]. The generation of scenarios is based on the change of the parameters of the model. The Vasicek model is a type of one factor model (short rate model) as it describes interest rate movements as driven by only one source of market risk. The model can be used in the valuation of interest rate derivatives. The model specifies that the instantaneous interest rate follows the stochastic differential equation:

$$dr_t = a(b - r_t)dt + \sigma dW_t$$

where W_t is a Wiener process modelling the random market risk factor. The standard deviation parameter σ determines the volatility of the interest rate. Vasicek's model was the first one to capture mean reversion, an essential characteristic of the interest rate that sets it apart from other financial prices. Thus, as opposed to stock prices for instance, interest rates can not rise indefinitely. This is because at very high levels they would hamper economic activity, prompting a decrease in interest rates. Similarly, interest rates can not decrease indefinitely. As a result, interest rates move in a limited range, showing a tendency to revert to a long run value. The drift factor $a(b - r_t)$ represents the expected instantaneous change in the interest rate at time t . The parameter b represents the long run equilibrium value towards which the interest rate reverts. Indeed, in the absence of shocks ($dW_t = 0$), the interest remains constant when $r_t = b$. The parameter a , governing the speed of adjustment, needs to be positive to insure stability around the long term value. For example, when r_t is below b , the drift term $a(b - r_t)$ becomes positive for positive a , generating a tendency for the interest rate to move upwards (toward equilibrium). The main disadvantage is that, under Vasicek's model, it is theoretically possible for the interest rate to become negative, an undesirable feature [21, 22].

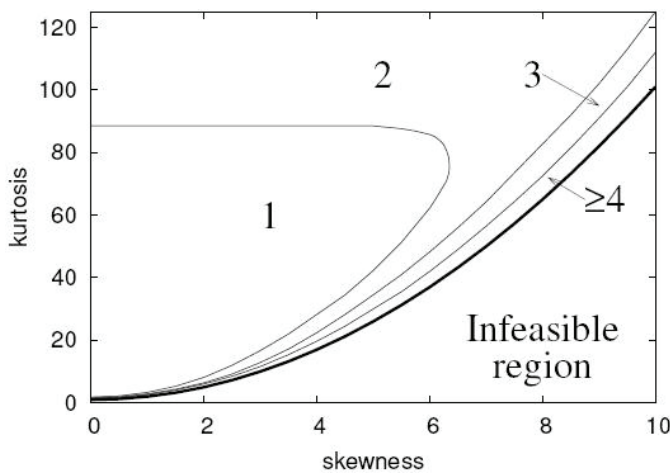


Fig. 3. Combinations of skewness and kurtosis accessible by a repeated cubic transformation of the standard normal distribution. The numbers show the number of cubic transformations needed to obtain distributions from the corresponding areas. The lowest region contains infeasible combinations of skewness and kurtosis [24].

3.1 Random sampling

In random sampling we sample from the error distribution of the vector autoregressive model. Given the estimated coefficients and the estimated covariance matrix of the vector autoregressive model, we can draw one random vector of yearly returns for bonds, real estate, stocks, deposits, wage growth, etc. If we would like to construct an event tree with ten nodes after one year (if we assume that the duration of each stage is

one year), we can simply repeat this procedure ten times, sampling independent vectors of returns for each node. The nodes at stage two in the event tree can also be sampled randomly, however the conditional distribution from stage one to stage two depends on the outcomes at the first stage. For example, wage growth follows an autoregressive process, so the expected wage growth from year one to year two depends on the realized wage growth rate in the previous period. An entire event tree for the stochastic program can be created by applying random sampling recursively, from stage to stage, while adjusting the conditional expectations of wage growth and deposits in each node based on previous outcomes [15].

The random sampling procedure for constructing a sparse multi-period event tree apparently leads to unstable investment strategies. An obvious way to deal with this problem is to increase the number of nodes in the randomly sampled event tree, in order to reduce the approximation error relative to the vector autoregressive model. However, the stochastic program might become computationally intractable if we increase the number of nodes at each stage, due to the exponential growth rate of the tree. Alternatively, the switching of asset weights might be bounded by adding constraints to the model or enforcing robustness through the choice of an objective function. Although we might get a more stable solution in this case, the underlying problem remains the same: the optimal decisions are based on an erroneous representation of the return distributions in the event tree [15].

3.2 Adjusted random sampling

An adjusted random sampling technique for constructing event trees can resolve some of the problems of the simple random sampling method. First, assuming an even number of nodes, we apply antithetic sampling in order to fit every odd moment of the underlying distribution. For example, if there are ten succeeding nodes at each stage then we sample five vectors of error terms from the vector autoregressive model. The error terms for the five remaining nodes are identical but with opposite signs. As a result we match every odd moment of the underlying error distributions (note that the errors have a mean of zero). Second, we rescale the sampled values in order to fit the variance. This can be achieved by multiplying the set of sampled returns for each particular asset class by an amount proportional to their distance from the mean. In this way the sampled errors are shifted away from their mean value, thus changing the variance until the target value is achieved. The adjusted values for the error terms are substituted in the estimated equations of the vector autoregressive model to generate a set of nodes for the event tree. Using adjusted random sampling to match the mean and the variance, we substantially reduce useless trading. The additional computational effort for adjusting the random samples is negligible. [15]

3.3 Conditional sampling

This is one of the most common methods for generating scenarios. At every node of a scenario tree, we sample several values from a stochastic process. This is done either by sampling directly from the distribution of the stochastic process, or by evolving the process according to an explicit formula. Traditional sampling methods can sample only from a univariate random variable. When we want to sample a random vector, we need to sample every marginal (the univariate component) separately, and combine them afterwards. Usually, the samples are combined all-against-all, resulting in a vector of independent random variables. The obvious problem is that the size of the tree grows exponentially with the dimension of the random vector: if we sample s scenarios for k marginals, we end-up with s^k scenarios. Another problem is how to get correlated random vectors. A common approach is to find the principal components (which are independent by definition) and sample those, instead of the original random variables. This approach has the additional advantage of reducing the dimension, and therefore reducing the number of scenarios. There are several ways to improve a sampling algorithm. Instead of a pure sampling, we may, for example, use integration quadratures or low discrepancy sequences, if appropriate. For symmetric distributions, an antithetic sampling can be used. Another way to improve a sampling method is to re-scale the obtained tree, to guarantee the correct mean and variance. For references to examples see [7].

3.4 Sampling from specified marginals and correlations

As mentioned in the previous section, the traditional sampling methods have problems generating multivariate vectors, especially if they are correlated. However, there are sampling-based methods that solve this problem, using various transformations. In those methods, the user specifies the marginal distributions and the correlation matrix. In general, there is no restriction on the marginal distributions; they may even be from different families. For references to examples see [7].

3.5 Moment matching

The methods from the previous section may be used only if we know the distribution functions of the marginals. If we do not know them, we may describe the marginals by their moments (mean, variance, skewness, kurtosis etc.) instead. In addition, we specify the correlation matrix and possibly – if the method allows us – other statistical properties (percentiles, higher comoments, etc). Then we construct a discrete distribution satisfying those properties. In Fig. 3 a map is showing the number of cubic transformations needed to achieve different combinations of skewness and kurtosis starting from standard normal distribution during moment matching.

Moment matching is a method to map a general distribution into a combination of exponential distributions. A popular approach in mapping a general probability distribution, G , into a phase type (PH) distribution, P , is to choose P such that some

moments of P and G agree. Matching the first moment of any nonnegative distribution is possible by a single exponential distribution. Matching the first moment is certainly important, but unfortunately it is often not sufficient, as ignoring the higher moments can result in misleading conclusions. Thus, it is desirable to match more moments of the input distribution G by P . Matching more moments may be possible if we are allowed to use many exponential distributions (phases). However, the use of many phases in the PH distribution increases the complexity of the Markov chain, and makes its analysis hard. Matching many moments using a small number of phases may be possible if we are allowed to use unbounded computational resources or if we limit the class of input distributions. However, these limitations are not desirable. Achieving all the four desirable properties is the challenge in designing a moment matching algorithm [23]:

- It is desirable that more moments of the input distribution, G , and the matching PH distribution, P , agree.
- It is desirable that P have a small number of phases.
- It is desirable that the algorithm be defined for a broad class of input distributions.
- It is desirable that the algorithm have short running time.

For references to examples see [7].

3.6 Path-based methods

These methods start by generating complete paths, i.e. the scenarios, by evolving a stochastic process. The result of this step is not a scenario tree, but a set of paths, also called a “fan” or trajectories. To transform a fan to a scenario tree, the scenarios have to be clustered (bound) together, in all-but-the-last period. This process is called clustering or bucketing. For references to examples see [7].

3.7 Optimal Discretization

Optimal discretization is a method that tries to find an approximation of a stochastic process (i.e. scenario tree) that minimizes an error in the objective function of the optimization model. Unlike the methods from the previous sections, the whole multi-period scenario tree is constructed at once. On the other hand, it works only for univariate processes. For references to examples see [7].

3.8 Simulation and randomized clustering based scenario tree generation approach

Paper [14] describes a procedure based on simulation and randomized clustering to generate the event tree which is the input to the financial optimization problems. The basic data structure is the scenario tree node, which contains a cluster of scenarios (vectors in R^n), one of which is designated as the centroid. The final tree consists of the centroids of each node, and their

branching probabilities. They introduce a randomized clustering algorithm which can be repeated until an acceptable clustering is found. The main steps of our algorithm can be outlined as follows:

- Step 1 (Initialization): Create a root node, with N scenarios. Initialize all the scenarios (including the centroid) with the desired starting point ('today's' prices). Form a job queue consisting of the root node.
- Step 2 (Simulation): Remove a node from the job queue. Simulate one time period of growth (from 'today' to 'tomorrow') in each scenario.
- Step 3 (Randomized seeds): Randomly choose a number of distinct scenarios around which to cluster the rest: one per desired branch in the scenario tree.
- Step 4 (Clustering): Group each scenario with the seed point to which it is the closest. If the resulting clustering is unacceptable, return to step 3.
- Step 5 (Centroid selection): For each cluster, find the scenario which is the closest to its center, and designate it as the centroid.
- Step 6 (Queuing): Create a child scenario tree node for each cluster (with probability proportional to the number of scenarios in the cluster), and install its scenarios and centroid. If the child nodes are not leaves, append to the job queue. If the queue is non-empty, return to step 2. Otherwise, terminate the algorithm.

For more details see [14].

3.8.1 Optimization based scenario tree generation approach

In an optimization approach to generate a scenario tree, the decision maker specifies the market expectations by the statistical properties that are relevant for the problem to be solved. The event tree is constructed so that these statistical properties are preserved. This is done by letting stochastic returns and probabilities in the scenario tree be decision variables in a non-linear optimization problem where the objective is to minimize the square distance between the statistical properties specified by the decision maker and the statistical properties of the constructed tree. The decision variables of the optimization problem are the prices (or returns) of a set of assets and probabilities of the event tree. Two alternative ways of applying the optimization approach are investigated in paper [14] to construct the event tree. If this scenario tree is constructed by considering the branching at each node separately, then we call it sequential optimization. In this case, a small non-linear optimization problem is constructed and solved at each node of the scenario tree. An alternative approach is to consider all nodes of the event tree and generate the whole tree in one large non-linear optimization problem, which we call overall optimization. For more details see [14].

3.9 Probabilistic neural networks

In a well-known paper from 1990, new neural network architecture was proposed called probabilistic neural network. By replacing the sigmoid activation function, often used in neural networks with exponential functions, a probabilistic neural network (PNN) is formed that can compute nonlinear decision boundaries. The PNN asymptotically approaches the Bayes optimal decision surface [10]. A probabilistic neural network can model the joint distribution of many discrete or continuous random variables [9]. PNNs are the synergistic mixture of Bayesian classifiers and Parzen window classifiers. The Bayes theory takes into account the relative likelihood of events and uses a priori information to improve prediction. Parzen estimators were developed to construct the probability density functions required by the Bayes theory. This approach provides an optimum pattern classifier in terms of minimizing the expected risk of wrongly classifying an object [17].

Probabilistic neural networks feature a feed-forward architecture and supervised training algorithm similar to back-propagation. But instead of adjusting the input layer weights using the generalized delta rule, each training input pattern is used as the connection weights to a new hidden unit. In effect, each input pattern is incorporated into the PNN architecture. This technique is extremely fast, since only one pass through the network is required to set the input connection weights. Additional passes might be used to adjust the output weights to fine-tune the network outputs. The network contains an input layer, which has as many elements as there are separable parameters needed to describe the objects to be classified. It has a pattern layer, which organizes the training set such that each input vector is represented by an individual processing element. Normally, there are equal amounts of processing elements for each output class. Otherwise, one or more classes may be skewed incorrectly and the network will generate poor results. And finally, the network contains an output layer, called the summation layer, which has as many processing elements as there are classes to be recognized [17].

Various clustering schemes have been proposed to cut down on the number of hidden units when input patterns are close in input space and can be represented by a single hidden unit. Given enough input data, the probabilistic neural network will converge to a Bayesian (optimum) classifier. Probabilistic neural networks allow true incremental learning where new training data can be added at any time without requiring retraining the entire network. PNNs are suitable to generate scenario trees since the output of the PNN is the assigned probability to each output class. These output classes with the corresponding probability values represent a single stage scenario tree where the classes are the branches of the scenario tree.

3.10 Internal sampling methods

Instead of using a pre-generated scenario tree, some methods for solving stochastic programming problems sample the sce-

narios during the solution procedure. The most important methods of this type are: stochastic decomposition, importance sampling within Benders' (L-shaped) decomposition, and stochastic quasigradient methods. For references to these methods see paper [7]. In addition, there are methods that proceed iteratively: they solve the problem with the current scenario tree, add or remove some scenarios and solve the problem again. Hence, at least in principle, the scenarios are added exactly where needed. The methods differ in the way they decide where to add/remove the scenarios. It is possible to use dual variables from the current solution, or to measure the "importance of scenarios" by EVPI (expected value of perfect information). [7]

3.11 Scenario reduction

Given a convex stochastic programming problem with a discrete initial probability distribution, the problem of optimal scenario reduction is stated as follows: Determine a scenario subset of prescribed cardinality and a probability measure based on this set that is closest to the initial distribution in terms of a natural (or canonical) probability metric. Arguments from stability analysis indicate that Fortet-Mourier type probability metrics may serve as such canonical metrics. Efficient algorithms are developed and described in paper [16] that determine optimal reduced measures approximately. Numerical experience is reported for reductions of electrical load scenario trees for power management under uncertainty. For instance, it turns out that after a 50% reduction of the scenario tree the optimal reduced tree still has about 90% of relative accuracy. For more details see [16]. Fig. 4 shows a simple illustration of scenario reduction.

The known scenario reduction algorithms determine a scenario subset (of prescribed cardinality or accuracy) and assign new probabilities to the preserved scenarios such that the corresponding reduced probability measure (the probability distribution of the approximation process) is the closest to the original measure (the probability distribution of the original empirical process) in terms of a certain probability distance between the original and the approximation probability distribution. The probability distance trades off scenario probabilities and distances of scenario values. As a frequently used distance, Kantorovich distance can be used for this purpose. This distance is the optimal value of a linear transportation problem. The interpretation of an optimal redistribution rule is that the new probability of a preserved scenario is equal to the sum of its former probability and of all probabilities of deleted scenarios that are closest to it with respect to the distance of the original and the approximation probability distribution. [3]

Paper [7] describes a method for decreasing the size of a given tree. This method tries to find a scenario subset of prescribed cardinality, and a probability measure based on this set, that is closest to the initial distribution in terms of some probability metrics [7].

A common approach in conditional sampling (see 2.5.) is to find the principal components (which are independent by defini-

tion) and sample those, instead of the original random variables. This approach has the additional advantage of reducing the dimension, and therefore reducing the number of scenarios [7].

3.12 Arbitrage constraints in financial scenario tree generation

Some financial applications, such as option pricing, require scenario trees which are free from arbitrage. [14] The absence of arbitrage opportunities is an important property for event trees of financial predictions (e.g. asset returns) that are used as input for stochastic programming models.

An arbitrage opportunity is a self-financing trading strategy that generates a strictly positive cash flow between 0 and T in at least one state and does not require an outflow of funds at any date. It is clear that investors would engage in such a trading strategy as much as possible if we assume that investors always prefer more to less. Therefore, such a trading opportunity cannot exist if market is in equilibrium. Formally, it is said that there are no arbitrage opportunities in the market if and only if there exists a unique risk-neutral (Martingale) probability measure [14].

If there is an arbitrage opportunity in the event tree, then the optimal solution of the stochastic programming model will exploit it. An arbitrage strategy creates profits without taking risk, and hence it will increase the objective value of nearly any financial planning model. So the presence of arbitrage opportunities in a portfolio optimization model can lead to substantial biases in the optimal solution that are due to profit opportunities which exist in the model. Even though the profit opportunities are often unlikely to materialize in reality, it is prudent for long term financial planning applications to generate scenarios that do not allow for arbitrage.

A potential problem for stochastic programming models in financial optimization is arbitrage opportunities in the event tree that are due to approximation errors. Arbitrage opportunities might arise because the underlying return distributions are sometimes approximated poorly with a small number of nodes in the event tree. If the application only involves broad asset classes such as a stock index, a bond index and real estate index, then arbitrage opportunities are unlikely to occur unless the errors in the event tree are very big. However, applications that involve options, multiple bonds or other interest rate derivative securities can be quite vulnerable to these problems. For example, the prices of European call and put options with equal strike price should satisfy put-call parity in each node of the event tree. If this relationship is violated because of a small approximation error, then the event tree contains an arbitrage opportunity and hence a source of spurious profits for the stochastic programming model. To deal with the arbitrage free problem, an aggregation method can be a solution. It starts with a very fine-grained event tree of asset prices without arbitrage opportunities and then reduces it to a smaller tree, while preserving the property of no-arbitrage. Recursively, a combination of nodes at a particular time period can be replaced by one aggregated node,

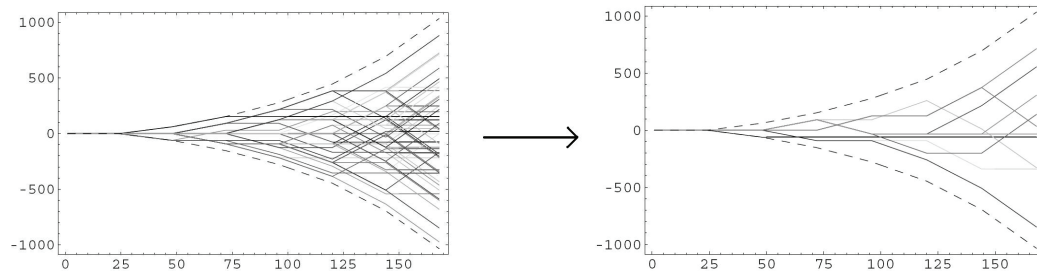


Fig. 4. Simple illustration of scenario reduction

while preserving the no-arbitrage property. If a node has only one particular successor remaining at the next time, then the intermediate period can be eliminated. This method can reduce the recombining lattice to a much smaller event tree with less trading dates, while meeting the arbitrage free condition. Another method for reducing a fine-grained lattice of security prices to a sparse event tree without arbitrage solves an option hedging problem with two sources of uncertainty: the stock price and stochastic volatility. First a three-dimensional fine-grained grid of time versus stock price and volatility is constructed to calculate option prices. Second, the points on the grid are partitioned into groups at a small number of trading dates, corresponding to the decision stages in the stochastic programming model. Each group of points on the grid is represented by a single aggregated node in the event tree of the stochastic programming model. If the prices in each aggregated node are calculated as a conditional expectation under the risk neutral measure of the prices in the corresponding partition on the grid, then the aggregated event tree will not contain arbitrage opportunities. Although the absence of arbitrage opportunities is important for financial stochastic programs with derivative securities, one should keep in mind that it is only a minimal requirement for the event tree. The fact that the stochastic program can not generate riskless profits from arbitrage opportunities does not imply that the event tree is also a good approximation of the underlying return process. We still have to take care that the conditional return distributions of the assets are represented properly in each node of the event tree. In order to avoid computational problems that arise if the tree becomes too big, one could reduce the number of stages of the stochastic program. In this way more nodes are available to describe the return distributions accurately. It is also important to include more nodes for the earlier stages, while larger errors in the later stages will have a small effect on the first-stage decisions which are the decisions implemented today by the decision makers. [15]

3.13 Considering empirical rules of time series

There is a long tradition of studies that try to explain the links between prices, interest rates, monetary policy, output and inflation. A class of linear-quadratic jump-diffusion processes was developed to describe the arbitrage-free time-series model of yields in continuous time that incorporates a country's central bank policy. A general framework for these models has been

recently developed, where certain factors are introduced to that influence the marginal productivity of capital, and thus the interest rates in the economy. For references see [1]. During scenario tree generation these kind of empirical rules can be useful to consider and integrate into the process.

4 Application examples

4.1 Optimal public debt management

The management of public debt is of paramount importance for any country. The Public Debt Management Division of the Italian Ministry of Economy decided to use scenario generation method to determine the composition of the portfolio issued every month that minimizes a predefined objective function that can be described as an optimal combination between cost and risk of the public debt service. The Italian Treasury Department issues about ten different types of securities including one with floating rate. Securities differ in the expiration date and in the rules for the payment of interests. Short term securities do not have coupons. Medium and long term securities pay cash dividends every six month, by means of coupons. The problem was to find a strategy for the selection of public debt securities that minimizes the expenditure for interest payments and satisfies, at the same time, the constraints of debt management. The time frame was 3-5 years in which the scenarios were generated. [1]

4.2 Power management

Paper [3] describes a case study applying scenario tree generation and scenario reduction for power management. Here, the stochastic process can be influenced by electricity load, stream flows to hydro units, fuel and electricity prices, etc. The scenario tree generator provided scenarios for a hydro-thermal generation system of a German institute. The optimization model determined the trading activities and the production decisions of the generation system such that the expected revenue was maximized. [3]

4.3 Pension fund asset liability management

Paper [5] describes a multistage stochastic programming model for a large Dutch pension fund to manage asset liability. Both randomly sampled event trees and event trees fitting the mean and the covariance of the return distribution were used for generating the coefficients of the stochastic program. The plan-

ning horizon of most pension funds stretches out for decades, as a result of the long term commitment to pay benefits to the retirees. However, funds should also reckon with short term solvency requirements. The trade-off between long term gains and short term losses should be made carefully, while anticipating future adjustments of the policy. This setting therefore seems to be suited for a stochastic programming approach with dynamic portfolio strategies. To fund the pension scheme, a plan sponsor pays a contribution to the fund each year. The pension fund had to decide how to invest these contributions in order to meet short term solvency requirements and to fulfill its long term obligations. Therefore the goal of asset liability management for the mentioned Dutch pension fund was to find a good investment policy and contribution policy. They used rolling horizon simulations to measure the performance of the stochastic programming model and the scenario generation methods they tried. They showed that the optimal solution of the stochastic program based on a randomly sampled tree promised spurious profits. They showed advanced methods based on adjusted random sampling that resulted in better results. [5]

4.4 Global capital market scenario generation system

Paper [6] describes a global capital market system based on scenario generation developed by Towers Perrin, one of the world's largest actuarial consulting companies. They implemented the scenario generation system based on a cascading set of stochastic differential equations. The system applies to financial systems for pension plans and insurance companies throughout the world. A case study is provided in the paper to illustrate the process. [6]

4.5 Predicting small business loan default with credit scoring

Paper [11] describes a method to predict small business loan default based on probabilistic neural network which generates scenario tree to support the decision making process. The methodology is used to construct and validate a model employing data from a pool of terminated small business loans. A total of 138 variables representing loan characteristics were initially examined and subsequently reduced to a set of five input variables that are effective predictors of loan default. These variables, which were composed mainly of traditional financial ratios, were then used to build a probabilistic neural network model that created a scenario tree with corresponding probabilities. The final probabilistic neural network correctly predicted the ultimate disposition of 92% of the loans in the out-of-sample testing. [11]

4.6 Predicting micro-loan defaults

Paper [12] describes a probabilistic neural based method to generate scenarios in the field of micro-loan defaults prediction. They show that probabilistic neural networks are effective in predicting loan defaults when the data is insufficient for use of

more traditional methods. PNN was found to be more effective than traditional statistical procedures including logit and discriminant analysis. From the initial 24 variables of micro loan applications, only seven were needed to predict 83% of defaulting and 88% of non-defaulting loans. In contrast, a logit model accuracy predicted only 47% of the default loans and 85% of the non-default loans. A discriminant model predicted 56% of the default loans and 85% of the non-defaulting loans. [12]

4.7 Multiple period scenario tree generation for optimal allocation of funds amongst main groups of asset classes

Paper [13] describes a method to generate multiple period scenario trees for optimal allocation of funds. They assumed that the decision maker is to split the funds amongst cash, bonds, domestic stocks and international stocks. They showed how to generate single period scenario tree and multiperiod scenario tree as well. They identified the critical success factors to use the model and drew attention to derived, implicit, over and under specifications and how to avoid these pitfalls. [13]

4.7.1 Risk management under the regulations of Basel II

Basel II is one of the biggest financial shake-ups in recent times, which will eventually lead to new rules and regulations for banking globally. Banks will need to have their processes and systems in place by the start of 2007, which is when the Basel Committee on Banking Supervision plans to implement the Accord. [18] To meet the requirements of Basel II, banks have to implement new countermeasures to reduce credit and operational risk. The pressure intensifies because banks have to ready by 2007. Scenario tree generation methods can help to meet the requirements of Basel II by quantifying and predicting the risk exposure of banks in forms of likely scenarios with corresponding probabilities.

Under the capital requirements of Basel II, it is important to develop an exact method to price loans, predict the probability of default and the loss given default, and monitor the risk exposure of the future. Paper [19] describes the loan pricing implications of Basel II.

The asymptotic single risk factor approach is a framework for determining regulatory capital charges for credit risk, and it has become an integral part of the second Basel Accord. Within this approach, a key parameter is the average asset correlation. Paper [20] examines the empirical relationship between firm probability of default and firm asset size measured by the book value of assets, using data from year-end 2000, credit portfolios consisting of US, Japanese, and European firms. The empirical results suggest that average asset correlation is a decreasing function of probability of default and an increasing function of asset size. The results suggest that these factors may need to be accounted for in the final calculation of regulatory capital requirements for credit risk. [20]. Scenario tree generation can be of help to not just provide point estimation but likely scenarios with assigned

probabilities such that the regulations of Basel II can be satisfied.

References

- 1 **Bernaschi M, Briani M, Papi M, Vergni D**, *Scenario Generation Methods for an Optimal Public Debt Strategy*, Quantitative Finance, 2004.
- 2 **Pflug G.Ch.**, *Scenario Tree Generation for Multiperiod Financial Optimization by Optimal Discretization*, 2004.
- 3 **Gröwe-Kuska N, Heitsch H, Römisch W**, *Scenario Reduction and Scenario Tree Generation for Power Management problems*.
- 4 **Heitsch H, Römisch W**, *Generation of Multivariate Scenario Trees to Model Stochasticity in Power Management*, 2003.
- 5 **Kouwenberg R**, *Scenario Generation and Stochastic Programming models for Asset Liability Management*, European Journal of Operational Research, vol. 131, Elsevier Science, 2001.
- 6 **Mulvey JM, Thorlacius AE**, *The Towers Perrin Global Capital Market Scenario Generation System*, 1998.
- 7 **Kaut M, Wallace SW**, *Evaluation of Scenario Generation Methods for Stochastic Programming*, 2003.
- 8 **Diane Reynolds**, *A Framework for Scenario Generation; ALGO Research Quarterly*, September 2001.
- 9 **Bengio Y**, *Probabilistic Neural Network Models for Sequential Data*, 2003.
- 10 **Specht DF**, *Probabilistic Neural Networks*, 1990.
- 11 **Yegorova I**, *A Successful Neural Network-based Methodology for Predicting Business Loan Default*, Credit & Financial Management Review (2001). Forth Quarter.
- 12 **De Lurgio SA**, *Predicting Micro-Loan Defaults using Probabilistic Neural Networks*, Credit & Financial Management Review (2002). Second Quarter.
- 13 **Hoyland K, Wallace S**, *Generating Scenario Trees for Multistage Decision Problems*, Management Science, vol. 47, February 2001.
- 14 **Gülpinar N, Rustem G, Settergren R**, *Simulation and Optimization Approaches to Scenario Generation*, Journal of Economic Dynamics & Control, vol. 28, Elsevier Science, 2004.
- 15 **Li-Yong Y, Xiao-Dong J, Shou-Yang W**, *Stochastic Programming Models in Financial Optimization*, Advanced Modelling and Optimization, vol. 5, 2003.
- 16 **Dupacová J, Gröwe-Kuska N, Römisch W**, *Scenario Reduction in Stochastic Programming: An Approach Using Probability Metrics*, Mathematical Programming, Ser. A95, 2003.
- 17 **Anderson D, McNeill G**, *Artificial Neural Networks Technology*, 1992. Kaman Sciences Corporation report.
- 18 **Porter D**, *Basel II: Heralding the Rise of Operational Risk*, Computer Fraud & Security, 2003.
- 19 **Repullo R, Saurez J**, *Loan Pricing under Basel II capital requirements*, Journal of Financial Intermediation, vol. 13, Elsevier Science, 2004.
- 20 **Lopez JA**, *The Empirical Relationship between Average Asset Correlation, Firm Probability of Default, and Asset Size*, Journal of Financial Intermediation, vol. 13, Elsevier Science, 2004.
- 21 **Hull JC**, *Options, Futures and Other Derivatives*. Upper Saddle River, NJ Prentice Hall, 2003.
- 22 **Vasicek O**, *An Equilibrium Characterisation of the Term Structure*, Journal of Financial Economics, vol. 5, 1977.
- 23 **Neuts MF**, *Probability distributions of phase type*. In *Liber Amicorum Professor Emeritus H. Florin*, University of Louvain, Belgium, 1975.
- 24 **Høyland K, Kaut M, Wallace SW**, *A Heuristic for Moment-matching Scenario Generation*, Computational Optimization and Applications, Vol. 24, 2003.
- 25 **Kaut M**, *Updates to the published version of A Heuristic for Moment-matching Scenario Generation*, 2003.